Egyptian Informatics Journal (2011) **12**, 45–51

Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

ORIGINAL ARTICLE

An extended k-means technique for clustering moving objects

Omnia Ossama, Hoda M.O. Mokhtar *, Mohamed E. El-Sharkawi

Faculty of Computers and Information, Cairo University Cairo, Egypt

Received 14 October 2010; accepted 18 January 2011
Available online 23 March 2011

KEYWORDS

Clustering moving objects;
Moving objects databases;
Mining moving object trajectories;
K-means clustering algorithm

Abstract k-means algorithm is one of the basic clustering techniques that is used in many data mining applications. In this paper we present a novel pattern based clustering algorithm that extends the k-means algorithm for clustering moving object trajectory data. The proposed algorithm uses a key feature of moving object trajectories namely, its direction as a heuristic to determine the different number of clusters for the k-means algorithm. In addition, we use the silhouette coefficient as a measure for the quality of our proposed approach. Finally, we present experimental results on both real and synthetic data that show the performance and accuracy of our proposed technique.

1. Introduction

With the fast advances in wireless and positioning technologies, we are currently facing a flood of location information. Today, we are capable of generating location information for mobile users, cars, buses, planes, animals, and other moving objects. This proliferation in location information along with the tremendous increase in the number of motor vehicles (an estimated increase of 3.69 million each year since 1960 [1]) increased the need for efficient location information management and analysis techniques. Hence, motivated by the fact that moving object data sets are usually huge in volume and complex in structure, efficient data mining algorithms and visual analysis techniques are thus required in order to extract useful and relevant information, and uncover regularities and patterns from this massive movement data sets. Consequently, a variety of disciplines including database research, m-commerce, transportation analysis, flight control systems, and animal migration behavior research show an increasing interest in mining moving objects' motion patterns [2]. For example, in the transportation context, with the proliferation in the number of vehicles on road networks, employing data mining techniques and specially clustering techniques in traffic control

* Corresponding author.
E-mail addresses: omnia@ieee.org (O. Ossama), h.mokhtar@fci-cu.edu.eg (H.M.O. Mokhtar), m.elsharkawi@fci-cu.edu.eg (M.E. El-Sharkawi).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.
doi:10.1016/j.eij.2011.02.007

Production and hosting by Elsevier

applications turn out to be a fruitful research direction. Mining and analyzing vehicle movement patterns could be used to predict traffic jams, send route congestion alerts, and present alternate route plans to travelers.

In this paper we propose a framework for clustering moving objects that employ the famous k-means clustering algorithm [3]. The framework is composed of 4 phases; computation phase, selection phase, clustering phase, and analysis phase. In brief, *in the computation phase*, the moving object database (MOD) is reconstructed to represent new features in the data set by evaluating motion direction properties for each object. Then, *in the selection phase* we select distinct sets of similar patterns using the output features from computation phase. Then, *in the clustering phase* we exploit the k-means algorithm. We use the output of the selection phase that is the number of dissimilar patterns to be the number of clusters for the k-means algorithm. In addition, inspired by the effect of initial centroid choice on the clustering quality and its impact in creating dead centroids; we propose to initialize the centroids based on trajectory dissimilarity. We initialize each of the k clusters by a virtual segment; its coordinate is the average position of all segments of trajectories that belongs to this cluster (pattern), and its direction is the direction of the majority of the segments in the cluster. Then, to ensure quality of resulting clustering we use the silhouette coefficient [4]. The silhouette coefficient is a measure for the clustering quality that is rather independent of the number of clusters k. Our main contributions are:

- Proposing an efficient clustering technique that overcomes the known problems of traditional k-means. The proposed technique uses a motion related heuristic to choose the optimal number of clusters and properly initializing the clusters centroids.
- Developing an accurate clustering algorithm by applying 2 similarity filters: *Euclidean distance* as a spatial measure, and a *direction based measure* as a shape-wise distance. We empirically prove both the efficiency and accuracy of the algorithm.
- Employing a novel approach for updating clusters' centroids to enhance the performance of our algorithm.
- Adjusting the silhouette coefficient calculation [4] to be computed in acceptable running time.

The rest of the paper is structured as follows: Section2 presents a literature survey of key related work. Section 3 defined the problem and presents our technical definitions. Section 4 presents our proposed clustering algorithm. Section 5 shows our experimental results. Finally, section 6 concludes the paper and proposes directions for future work.

2. Related work

There has been considerable research in the area of mining spatiotemporal data [5,6]. Clustering analysis has become an attractive research area and many successful approaches have been proposed. The generic definition of clustering is usually redefined depending on the type of data to be clustered and the clustering objective. Different and scalable clustering algorithms have been proposed in [7]. The k-means method is a widely used clustering technique; there have been a number of research works that describe competitive algorithms for the k-means problem [8–10]. Authors in [9] proposed a technique to seed the initial centers for k-means. Their idea is based on the intuition of spreading the k initial cluster centers away from each other, the firs cluster center is chosen uniformly at random from the data points that are being clustered, after which each subsequent cluster center is chosen from the remaining data points with probability proportional to its distance squared to the point's closest cluster center. Adaptive k-means is another proposed extension for the traditional k-means. Adaptive k-means clustering technique aims to overcome the dependence of traditional k-means on the choice of the number of clusters, and on the initialization of the centroids. The authors in [10] propose an algorithm to solve both issues. Our work is very close to the partition and group framework proposed in [11] which partitions a trajectory into a set of line segments, and then, groups similar line segments together into a cluster.

The primary advantage of this framework is to discover common sub-trajectories from a trajectory database. But in our work we follow the same path but we add the idea of defining trajectories' similarity based on the spatial distance (i.e., Euclidean distance) and movement direction to guarantee clustering trajectories that have the same movement direction and close to each other.

Inspired by the applications of clustering trajectories in air traffic systems; authors in [12] present two trajectory clustering methodologies that enable obtaining frequently flown routes. They use recorded radar tracks from a terminal area. Their objective was monitoring the instantaneous health of the airspace. They assume that the airspace is healthy when all aircrafts are flying according to the nominal procedures, and if it does not require more attention from the air traffic controller. Moreover, trajectory clustering could be used for learning moving objects behaviors [13]. In this paper the authors evaluate different similarity measures and clustering methodologies to catalog their strengths and weaknesses when utilized for the trajectory learning problem.

3. Problem definition

In this paper we focus on clustering moving objects trajectories. The aim of this work is to present an efficient moving object trajectory clustering algorithm along with constructing an efficient mechanism for computing the optimal number of clusters for the k-means clustering algorithm and properly initializing clusters' centroids. We measure trajectory similarity using two filters to ensure the accuracy and quality of resulting clusters. The first filter sorts the segments of different trajectories among clusters based on segments orientation (direction). The second filter refines the cluster members through applying the Euclidean distance [14] to measure the deviation between each new cluster member candidate and the current cluster centroid. If the deviation (distance measure) exceeds certain threshold then a new cluster with same orientation but different centroid is generated.

A trajectory of a moving object is typically modeled as a sequence of consecutive locations in a multidimensional Euclidean space (usually 2-dimensional space with time treated as a third dimension). In other words, a trajectory is a piece wise linear function of time. The main characteristic in our model

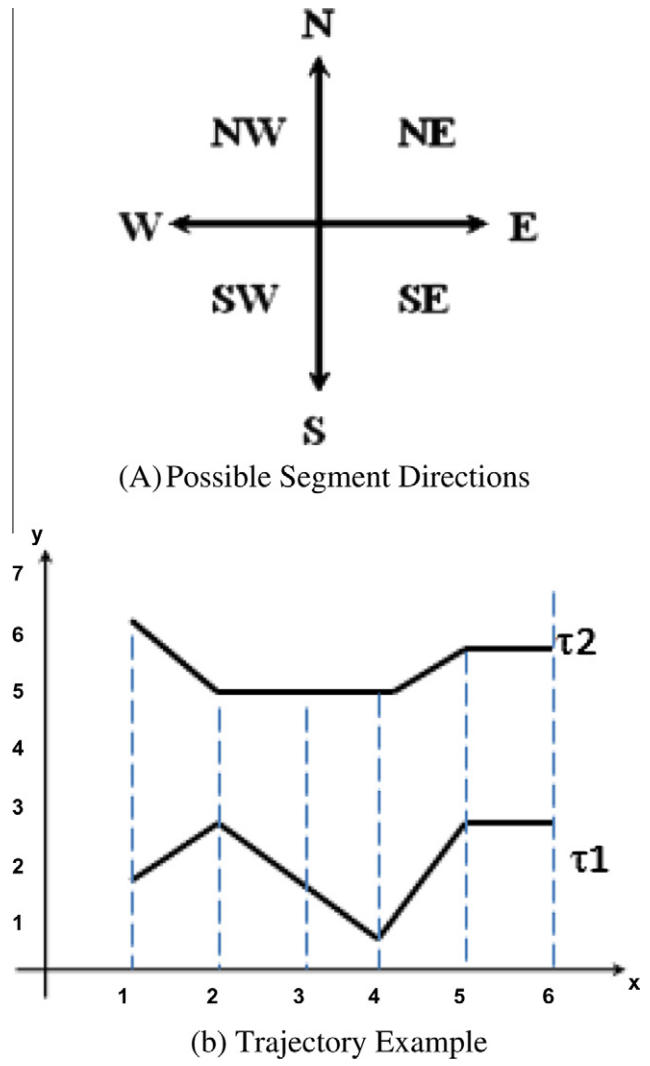

Figure 1 Illustrating segment direction computation.

is that we focus on an important aspect of a trajectory, namely, the direction of each trajectory segment. Let D be the finite set E,N,S,W,NE,NW,SE,SW, where 'E' is for the EAST direction, 'N' for the NORTH, 'S' for SOUTH, 'W' is for WEST, 'NE' is for NORTH-EAST, 'NW' is for NORTH-WEST, 'SE' is for SOUTH-EAST, and 'SW' is for SOUTH-WEST.

A segment direction 'd' is defined over the domain $\mathscr{D}$ as shown in Fig. 1(a).

Definition. 3.1. A trajectory segment is a continuous directed line segment represented by the triple $((x_i, y_i, t_i), (x_j, y_j, t_j), d_{ij})$, where (x_i, y_i, t_i) and (x_j, y_j, t_j) are the start and end positions , respectively, for the segment s_{ij} at time instances t_i and t_j, such that $i \leqslant j$, and d_{ij} is the direction of segment s_{ij}.

Definition. 3.2. A trajectory is a finite sequence of segments. A trajectory τ of length n is represented by the ordered list $(s_1, \ldots, s_n)$, where n is the number of segments in τ .

In our model we allow segments of same trajectory to be defined over different time intervals, however, we require corresponding segments in different trajectories to be defined over the same time interval.

Example. 3.3. Let MOD be a moving object database consisting of the two trajectories:

$\tau 1 = (s_{11}, s_{12}, \ldots, s_{1n})$ and $\tau 2 = (s_{21}, s_{22}, \ldots, s_{2n})$ shown in Fig. 1(b).The segments in each trajectory can have different lengths (i.e., defined over different time intervals), however, $|s_{11}| = |s_{21}|$, $|s_{12}| = |s_{22}|$, ... , $|s_{1n}| = |s_{2n}|$.

Definition. 3.4. A trajectory direction $(d_1, d_2, \ldots, d_n)$ is a list of directions for each segment of the trajectory, where, d_i is the direction of the ith segment, and each d_i D.

Having a model for moving object trajectories, in the following discussion we elaborate our notion of clustering moving object trajectories. As known clustering is simply a grouping of similar entities. In this paper we refer to a cluster as a set of similar trajectory segments such that in-cluster segments are both spatially close to each other according to a distance measure and share the same movement direction (spatial orientation). To create clusters, simply a trajectory is partitioned into its composing segments, then clustering is applied over those segments. Thus, a single trajectory can belong to multiple clusters based on its segments' clusters. Having cluster members to be trajectory segments, the cluster centroid is consequently a segment as well.

Definition. 3.5. Given a cluster C with centroid (i.e., segment) $c = ((x_{ci},y_{ci},t_{ci}),(x_{cj},y_{cj},t_{cj}),d_c)$, and a trajectory segment $s = ((x_{si},y_{si},t_{si}),(x_{sj},y_{sj},t_{sj}),d_s)$, where d_c and d_s are the centroid and segment direction, respectively.

The similarity distance between c and s is defined by sum of the Euclidean distance and directional evaluation between the segments as shown in Eq. (1).

$$D(c,s) = \text{Euclidean distance}(c,s) + \text{directional evaluation}(d_c, d_s) \quad (1)$$

where

$$\text{Euclidean distance } (c,s) = \frac{\int_t \mathrm{d}(x_c t + y_c, x_s, t + y_s)}{t}\,\mathrm{d}t$$

$D(x_c t + y_c, x_s t + y_s)$ is the Euclidean distance between line segments c and s.

$D((x_c, y_c), (x_s, y_s)) = \sqrt{(x_c - x_s)^2 + (y_c - y_s)^2}$ is the Euclidean distance between c, s at each time instant t. And

$$\text{Directional evaluation}(d_c, d_s) = \begin{cases} 0, \text{if } d_c \text{ and } d_s \text{ are both equal} \\ 1, \text{if } d_c \text{ and } d_s \text{ are different in 1 character} \\ 2, \text{otherwise} \end{cases}$$

Definition. 3.6. A trajectory cluster is a linked list of clusters $(c_1, c_2, \ldots, c_n)$ Such that, c_i is the cluster of the ith segment of the trajectory.

The remainder of the paper presents the proposed framework that employs the k-means clustering algorithm to group similar trajectory segments. Our goal is to present a novel approach to enhance trajectory clustering using the k-means algorithm through optimally choosing the initial number of clusters, calculating trajectories' similarity based on spatial distance and movement direction which support different kinds of applications like applications of clustering trajectories in air traffic systems [12].

4. Proposed extended k-means algorithm

Clustering is a key data mining task that aims to partition a given set of objects into groups (classes or clusters) such that objects within a cluster would have high degree of similarity to each other and low similarity to objects in other clusters [15]. In this paper, we propose a framework for clustering moving object trajectories. The proposed approach is based on the widely used k-means clustering algorithm. As the k-means algorithm seeks to minimize the average squared distance between points in the same cluster, our technique also seeks to group trajectories featuring similar motion pattern. However, in our technique we aim to overcome a major drawback of the k-means algorithm namely the assumption that number of clusters and initial clusters' centroids are given. This assumption is a crucial input for the k-means algorithm that affects both the algorithm performance and accuracy. Besides, proper cluster initialization is a major step to avoid the occurrence of dead centroids. Dead centroid problem is usually a consequence of poor cluster initialization that results in empty clusters being generated.

To overcome the above problems we use a heuristic to choose the number of clusters. The heuristic employed is based on the different motion patterns of the trajectories' segments in the data set. Next, we initialize each cluster with a virtual trajectory segment that we generate such that its motion direction is similar to the cluster's segments direction, and its spatial position is the average of the cluster's segments. The proposed framework is composed of 4 phases; *computation; selection; clustering, and analysis* that we discuss in more details in the remainder of this section.

1. Computation Phase: This phase is basically a pre-processing phase that is performed on the data set to discover the different motions patterns appearing in the data set and thus initialize a corresponding number of clusters. In this phase we first decompose each trajectory in the moving object database MOD into its constituting segments, and represent the trajectory as an ordered list of those segments. Next, we compute the direction of each segment using the *Compute Direction* procedure in Fig. 2. The output of this phase is a list of pairs associating each segment to its direction that belongs to the domain of directions D defined earlier. In the remainder of our discussion we will refer to this list as direction list.
2. Selection Phase: After computing the direction list in the computation phase. We count the number of distinct directions traveled by our trajectory data set. We use this number as a heuristic initialization for the number of clusters in our algorithm. Note that at this stage the number of possible clusters is at most 8 representing the universe of D.
3. Clustering Phase (E-km: Extended k-means): Here, k-means clustering method is exploited. Having an initial number of clusters from the previous step, we now apply another similarity measure to refine our cluster members. This step will thus increase the accuracy obtained from our algorithm. In this refinement stage we start adding segments to corresponding clusters based on the segment direction (orientation). We construct a cluster centroid as a virtual segment that has the same spatial orientation as the cluster orientation, and spatial position being the average of the cluster members so far. As we insert new segments to cluster we perform 2 steps:
   (a) Compute the Euclidean distance between the cluster centroid and the new segment, based on the resulting value the segment is either inserted into the cluster, or is tested against other existing cluster with the same orientation, or generates a new cluster with the new segment as the initial centroid.
   (b) The centroid of the cluster to which the segment is inserted is recalculated to take the new segment position into account. The new centroid is thus a virtual segment with same orientation as other segments in the cluster, and its position is the average of the spatial positions of the remaining cluster members.

The details of the algorithms developed in this phase are illustrated in Figs. 3 and 4.

4. Analysis Phase: In this phase we analyze the accuracy of the resulting clustering. We employ the silhouette coefficient as a measure for the clustering quality. In order to measure the clustering quality independent from the features used for clustering and the number of clusters produced as a result, our analysis uses the silhouette coefficient introduced in [4]. To evaluate the quality of a clustering we compute the average silhouette coefficient of all segments in each cluster. The silhouette coefficient provides a reliable quality measure, unfortunately calculating the silhouette coefficient takes quadratic time. Thus, we adjusted the silhouette coefficient calculation to be done in acceptable running time. We calculate silhouette coefficient based on the distance to cluster centroid instead of all segments in cluster. This simplification in the computation is still acceptable as the cluster centroid is a segment that we generated to represent the average of all other segments in the cluster. Hence, the segment to centroid distance is a fair representative for the distance between the segment and other clusters' members.

Definition. 4.1 (Silhouette Coefficient):. Let $C = (C_1, C_2, \ldots, C_m)$ describe an E-km clustering results. The distance of segment $s_i \in C_j$ is the minimum Euclidean distance between s_i and centroid of C_j.

$$Dist(s_i, C_m) = \min(\text{Euclidean distance}(s_i, C_i : centroid))$$

And the distance of segment s_i to other clusters C_m is the minimum of Euclidean distance to all other clusters that do not contain s_i.

$$Dist(s_i, C_m) = \min(\text{Euclidean distance}(s_i, C_m : centroid))$$

Then,

$$\text{Silhouette}(s_i) = \frac{Dist(s_i, C_j) - Dist(s_i, C_m)}{\max(Dist(s_i, C_j) - Dist(s_i, C_m))}$$

The value of the silhouette coefficient of a segment varies between -1 and $+1$. A value near -1 indicates that the segment is clustered badly. A value near $+1$ indicates that the segment is well clustered. Using the simplified computation procedure, the time for computing the silhouette coefficient is tremendously reduced.

Procedure: Compute_Direction(S: Segment)

Output: d_s: identifier of direction of segment S
if $(S.start_x \leqslant S.end_x$ *and* $S.start_y = S.end_y)$ **then**
d_s = N;
else if $(S.start_x \geqslant S.end_x$ *and* $S.start_y = S.end_y)$ **then**
d_s=S;
else if $(S.start_x = S.end_x$ *and* $S.start_y \leqslant S.end_y)$ **then**
d_s=E;
else if $(S.start_x = S.end_x$ *and* $S.start_y \geqslant S.end_y)$ **then**
d_s=W;
else if $(S.start_x \leqslant S.end_x$ *and* $S.start_y \leqslant S.end_y)$ **then**
d_s=NE;
else if $(S.start_x \leqslant S.end_x$ *and* $S.start_y \geqslant S.end_y)$ **then**
d_s=SE;
else if $(S.start_x \geqslant S.end_x$ *and* $S.start_y \leqslant S.end_y)$ **then**
d_s=NW;
else if $(S.start_x \geqslant S.end_x$ *and* $S.start_y \geqslant S.end_y)$ **then**
d_s=SW;
Returnd_s;

Figure 2 Computation phase.

Algorithm: Extended k-Means (E-km)(S, C)

Input: S: List of segments in MOD, C Initialized k cluster centroids, δ: Given threshold
Output: C_{List}: List of Clusters
foreach $(s \in S)$ **do**
  foreach $(c \in C)$ **do**
    TempDist=Direction Evaluation$(s.direction, c.direction)$ + EuclideanDistance(s, c);
  end
  MinDistance=Min[TempDist].centroid;
  ClosestCentroid=Min[TempDist].centroid;
  if $(MinDistance \leqslant \delta)$ **then**
  Cluster=C[ClosestCentroid];
  C_{List} = Update_Centroid$(Cluster, s)$;
  else
  $Cluster_New.centroid$= s ;
  C.Add$(Cluster_New.centroid)$;
end
return C_{List} ;

Figure 3 Clustering phase.

5. Experimental evaluation

We evaluated our algorithms on both real and synthetic data sets. The real data set represents the movement of 11 moving object in London during a one month period (July 2007) with a 10 sec sampling rate [16]. However, since the data set is relatively small (11 trajectories and 9498 segments), it could not expose the actual performance of the algorithms; therefore, the performance experiment was conducted using synthetic data set generated by the Brinkoff generator which is commonly used for generating realistic moving objects [17]. Using the generator, we simulated two dimensional trajectories of vehicles on the road network in the city of San Francisco. We measure the performance in terms of query processing time, and since all test datasets are relatively small in size, they fit into memory and no index structure is being considered. All

Procedure: Update_Centroid(C,NewSeg)

Input: C:is the selected cluster, $NewSeg$:new segment to be inserted in C
Output: C: Updated Cluster
C.Insert($NewSeg$);
C.centroid.x=Avg($c.AllSegments.x$);
C.centroid.y=Avg($c.AllSegments.y$);
C.centroid.direction=$NewSeg.direction$;
return C;

Figure 4 Updating centroid procedure.

experiments are conducted on a 2 GHz with 4 Gbyte of main memory.

In the first experiment we study the effect of changing similarity percentage of matching directions (movement patterns) on the number of clusters k with real and synthetic datasets. Basically, we vary the direction matching similarity percentage from 9% to 5% (i.e., 5% similarity means that two trajectories coincide (belong to the same movement pattern) if half of their segments have the same movement direction). Fig. 5 shows that the increase in the similarity percentage allows more objects to be grouped together. Consequently, the number of clusters (k) decreases as well. The difference between performance of E-km and traditional k-means on the synthetic data sets is substantial.

The second experiment studies the effect of changing segment direction similarity percentage among trajectories on the clustering computation time for both the E-km and k-means algorithms. The goal of this experiment is to ensure that our approach performs well in all circumstances (large data sets, large k, different similarity percentages, and different number of movement patterns). The experiment shows that due to the decrease in the number of clusters as a result of increasing similarity percentage, k-means takes more time in clustering all objects. This is due to the fact that small number of clusters means an increase in cluster size (i.e., number of segments per cluster). Therefore, it takes more time in each iteration to update selected cluster centroids. Nevertheless, E-km algorithm performs well in such case because updating virtual segment takes no time as it only requires the computation of the spatial mean between current centroid and newly added segment. Fig. 6 proofs that the performance gain is drastic on large synthetic data sets, besides the clustering obtained by E-km is obtained in almost half the time needed by the k-means.

Then, we measure the quality and accuracy of our proposed E-km algorithm through counting the number of dead centroids in the resulting clustering. In this test we use the silhouette coefficient [4] that tells if segments are well clustered or not. We compute the average silhouette coefficient of a cluster by simply taking the average silhouette coefficient of segments belonging to the cluster.

Fig. 7 shows that all silhouette values are positive.

Finally, Fig. 8 shows that E-km creates zero dead centroid in both synthetic and real data set in all dataset size (in this experiment we change dataset set sizes and consequently that affects on the generated number of clusters (k)) in contrast to k-means that randomly initializes centroids, and hence it creates empty clusters. E-km algorithm shows positive silhouette values and does not create empty clusters due to the prop-

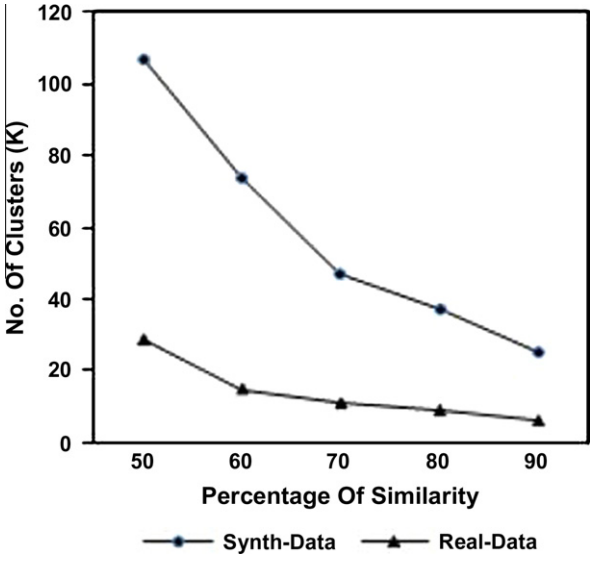

Figure 5 Accuracy of E-km algorithm.

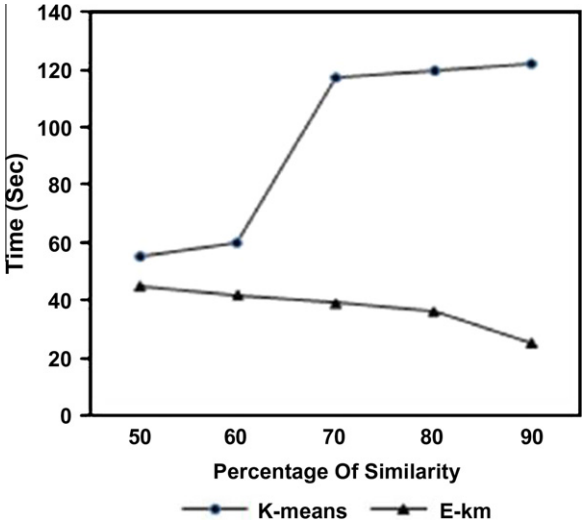

Figure 6 Performance of E-km vs. k-mean.

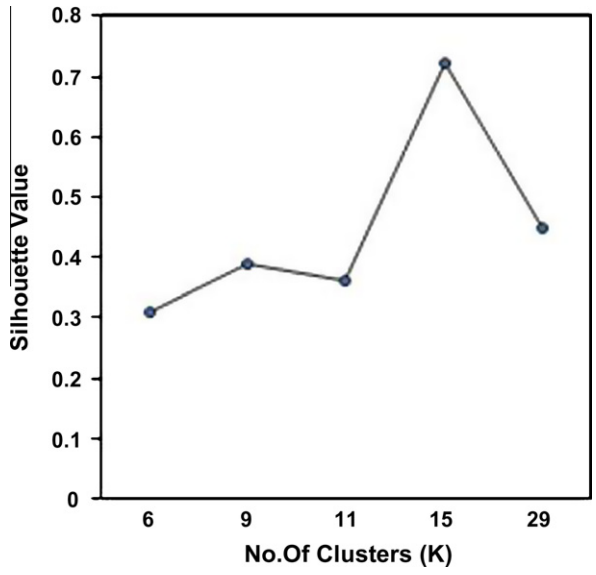

Figure 7 Average Silhouette value vs. number of clusters (k).

		E-km	k-means
	k	Number of dead centroids	Number of dead centroids
Real Dataset 1000	7	0	1
2000	9	0	2
3000	15	0	2
Synthetic Dataset 10000	17	0	2
20000	29	0	4
30000	50	0	7

Figure 8 Accuracy of E-km vs. k-means.

er initialization of centroids and properly choosing k based on data set properties (number of dissimilar patterns).

6. Conclusions and future work

In this paper we propose a pattern-based clustering algorithm that extends the k-means algorithm for trajectory data (E-km: Extended k-means). E-km approach overcomes the known drawbacks of the k-means algorithm, namely, the dependence on the number of clusters (k), and the dependence on the initial choice of the clusters' centroids. Moreover, our approach guarantees creating high accurate clusters. We measured accuracy of our approaches using silhouette coefficient. For future work, we believe that further improvements in the heuristic that we use to compute k are still applicable (i.e using the angular value for measuring movement patterns). We also plan to compare our proposed algorithm to other k-means clustering techniques. Also, research on clustering moving objects is still an open direction.

References

[1] Highway statistics 2008, < http://www.fhwa.dot.gov/policyinformation/statistics/2008/mv10.cfm > (February 2010).
[2] < http://www.environmental-studies.de/projects/projects.html > (December 2009).
[3] Macqueen JB. Some methods of classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, 1967. p. 281–97.
[4] Kaufman L, Rousseeuw P, editors. Finding groups in data: an introduction to cluster analysis. New York: Wiley; 1990.
[5] Buchin K, Buchin M, Gudmundsson J, Löffler M, Luo J. Detecting commuting patterns by clustering subtrajectories. In: ISAAC '08: Proceedings of the 19th international symposium on algorithms and computation. Berlin, Heidelberg: Springer-Verlag; 2008. p. 644–655.
[6] Yiu ML, Mamoulis N. Clustering objects on a spatial network. In: SIGMOD' 04: Proceedings of the 2004 ACMSIGMOD international conference on management of data. New York, NY, USA: ACM; 2004. p. 443–54.
[7] Liu W, Wang Z, Feng J. Continuous clustering of moving objects in spatial networks. In: KES '08: Proceedings of the 12th international conference on knowledge-based intelligent information and engineering systems, Part II. Berlin, Heidelberg: Springer-Verlag; 2008. p. 543–50.
[8] Frahling G, Sohler C. A fast k-means implementation using coresets. In: SCG '06: Proceedings of the twenty-second annual symposium on computational geometry. New York, NY, USA: ACM; 2006. p. 135–43.
[9] Arthur D, Vassilvitskii S. k-means + +: the advantages of careful seeding. In: SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics; 2007. p. 1027–35.
[10] Hailin C, Xiuqing W, Junhua H. Adaptive k-means clustering algorithm, MIPPR 2007. Pattern Recognition and Computer Vision, 2007.
[11] Lee J-G, Han J, Whang K-Y. Trajectory clustering: a partition-and-group framework. In: SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on management of data. New York, NY, USA: ACM; 2007. p. 593–604.
[12] Gariel M, Srivastava AN, Feron E. Trajectory clustering and an application to airspace monitoring, ArXiv e-prints.
[13] Morris BT, Trivedi MM. Learning trajectory patterns by clustering: Experimental studies and comparative evaluation. In: Proceedings of IEEE conference on Computer Vision and Pattern Recognition, 2009.
[14] Juan-chico J, Bellido MJ, Acosta AJ, Barriga A. Efficient similarity search in sequence databases. In: Foundations of data organization and algorithms 1993:69–84.
[15] Sofia Moldovan GS. Aspect mining using a vectorspace model based clustering approach. In: AOSD'06: aspect-oriented software development conference, Bonn, Germany, 2006.
[16] http://www.ecourier.co.uk, Aug 2010.
[17] Brinkhoff T. Generating traffic data. IEEE Comput Soc Tech Committe Data Eng 2003:19–25.